

RADC-TR-89-100
Final Technical Report
August 1989



AD-A211 849

FAULT-TOLERANT SECURE SYSTEM EVALUATION FOR NON-VON NEUMANN ARCHITECTURE

Syracuse University

Sponsored by
Strategic Defense Initiative Office



APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Strategic Defense Initiative Office or the U.S. Government.

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

89 9 01 05 2


This report has been reviewed by the RADC Public Affairs Division (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-89-100 has been reviewed and is approved for publication.

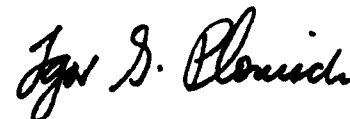
APPROVED:


ALAN N. WILLIAMS, 1Lt, USAF
Project Engineer

APPROVED:


RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command & Control

FOR THE COMMANDER:



IGOR G. PLONISCH
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COTC) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

FAULT-TOLERANT SECURE SYSTEM EVALUATION
FOR NON-VON NEUMANN ARCHITECTURE

C. V. Ramamoorthy

Contractor: Syracuse University
Contract Number: F30602-81-C-0169
Effective Date of Contract: 10 June 1987
Contract Expiration Date: 31 December 1987
Short Title of work: Fault-tolerant Secure System Evaluation
for Non-Von Neumann Architecture
Period of Work Covered: Jun 87 - Dec 87

Principal Investigator: C. V. Ramamoorthy
Phone: (415) 642-4751

RADC Project Engineer: Alan N. Williams, 1Lt, USAF
Phone: (315) 330-2925

Approved for public release; distribution unlimited.

This research was supported by the Strategic Defense
Initiative Office of the Department of Defense and
was monitored by 1Lt Alan N. Williams, RADC (COTC)
Griffiss AFB NY 13441-5700 under Contract
F30602-81-C-0169.



| | |
|--------------------|--|
| Accession For | |
| NTIS CRA&I | <input checked="checked" type="checkbox"/> |
| DTIC TAB | <input type="checkbox"/> |
| Unannounced | <input type="checkbox"/> |
| Justification | |
| By | |
| Distribution/ | |
| Availability Codes | |
| Dist | Availability or Special |
| A-1 | |

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

| REPORT DOCUMENTATION PAGE | | | | Form Approved OMB No. 0704-0188 | |
|---|-------|--|---|--|-----------------------------------|
| 1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED | | | 1b. RESTRICTIVE MARKINGS N/A | | |
| 2a. SECURITY CLASSIFICATION AUTHORITY N/A | | | 3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited. | | |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A | | | | | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S) N/A | | | 5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-89-100 | | |
| 6a. NAME OF PERFORMING ORGANIZATION Syracuse University | | 6b. OFFICE SYMBOL (if applicable) | 7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (COTC) | | |
| 6c. ADDRESS (City, State, and ZIP Code) Office of Sponsored Programs Skytop Office Bldg, Skytop Road Syracuse NY 13210 | | | 7b. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700 | | |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION Strategic Defense Initiative Office | | 8b. OFFICE SYMBOL (if applicable) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-81-C-0169 | | |
| 8c. ADDRESS (City, State, and ZIP Code) Office of the Secretary of Defense Wash DC 20301-7100 | | | 10. SOURCE OF FUNDING NUMBERS | | |
| | | | PROGRAM ELEMENT NO. 63223C | PROJECT NO. B413 | TASK NO. 03 |
| 11. TITLE (Include Security Classification) FAULT-TOLERANT SECURE SYSTEM EVALUATION FOR NON-VON NEUMANN ARCHITECTURE | | | | | |
| 12. PERSONAL AUTHOR(S) C. V. Ramamoorthy | | | | | |
| 13a. TYPE OF REPORT Final | | 13b. TIME COVERED FROM Jun 87 TO Dec 87 | | 14. DATE OF REPORT (Year, Month, Day) August 1989 | |
| 15. PAGE COUNT 32 | | | | | |
| 16. SUPPLEMENTARY NOTATION C. V. Ramamoorthy is with Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley. | | | | | |
| 17. COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Fault-tolerant, Computer security, Non-Von Neumann architecture, Parallel, Multiprocessor | | |
| FIELD | GROUP | SUB-GROUP | | | |
| 12 | 07 | | | | |
| 19. ABSTRACT (Continue on reverse if necessary and identify by block number) This effort was on the analysis of fault-tolerance and security in non-Von Neumann machines, including defining fault-tolerance and security, and surveying the trade-offs between security and performance, fault-tolerance and performance, and security and fault-tolerance. Fundamental issues regarding fault-tolerance and security are brought up and during the survey, various proposed formal models are studied, and important issues on practicality and trade-offs are examined. While the discussions on trade-offs are especially topical to non-Von Neumann machines, they are also pertinent for Von Neumann machines. | | | | | |
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS | | | 21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED | | |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL Alan N. Williams, 1Lt, USAF | | | 22b. TELEPHONE (Include Area Code) (315) 330-2925 | | 22c. OFFICE SYMBOL RADC (COTC) |

DD Form 1473, JUN 86

Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

1. Introduction

A major research effort was made during this reporting period on analysis of fault-tolerance and security in non-von Neumann machines, including defining fault-tolerance and security, and surveying the trade-offs between security and performance, fault-tolerance and performance, and security and fault-tolerance. Fundamental issues regarding fault-tolerance and security are brought up, and during the survey, various proposed formal models are studied. and important issues on practicality and trade-offs are examined. While the discussions on trade-offs are especially topical to non-von Neumann machines, they are also pertinent for von Neumann machines.

2. Definition of fault tolerance:

In this section, we first define and clarify the terms we will use in this report.

Definition of fault tolerance : *Fault tolerance is defined as ability of a system to continue correct operation in the presence of fault.*

Fault tolerant computing can be achieved by protective redundancy. Typical redundancy techniques are:

- (1) Hardware redundancy by additional circuit
- (2) Software redundancy by additional programs
- (3) Information redundancy by additional data
- (4) Temporal redundancy by repetition of operations

For practical purposes, additional constraints are introduced; for instance, the amount of extra execution time must be kept within reasonable bounds in real time fault tolerant systems. The redundant hardware and software introduced in the fault tolerant system must be kept at a minimum so that the total storage capacity and resources required must be within feasible limits. Even though we defined the term *fault tolerance* above, the term is used to represent different things for different systems. and by different people.

To clarify the definition of fault tolerance and to compare or measure degree of fault tolerance, we need a clear spectrum of metrics. Some of the metrics are as

follows.

In the total system level, the following metrics can be defined.

- (1) *Reliability*: It is defined as the probability that a system will operate correctly for a specified period of time, or in terms of the expected time between failures.
- (2) *Availability*: The probability that a system is operational at any given time.
- (3) *Safety*: The probability that a system does not get into a disastrous state regardless to its mission.

System designers must first clarify the goal of a system. Highly reliable systems require most stringent requirements; one example of these systems is embedded computer for defense system. Highly available systems, on the other hand, need less stringent requirements, and they permit short period of down-time and short period of failure should not develop into disastrous results. Highly safe systems are primarily concerned with the safety of people involved; one example of this system is a nuclear plant control computer.

Highly available systems usually does not cause safety problem, though crash of the system may cause some inconvenience to people; yet in designing a computer for nuclear plant control, the first priority is given to safety rather than reliability.

When we design fault tolerant, secure, and high performance Non-Von Neumann, we encounter numerous problems, and obviously there are trade-offs between them. At the processor level, we are generally interested in the reliability of systems. When we evaluate the machine, the following are fundamental issues we must consider.

Architectures: Obviously the architecture will affect fault tolerance. Popular non-Von Neumann architectures are SIMD (single-instruction, multiple-data-stream) computers, MIMD (multiple-instruction, multiple-data-stream) computers, and systolic arrays, and wavefront arrays.

The number of processors: The processors can be devoted either for high throughput or redundancy. If all of the processors available are devoted to pure computation, the system does not have redundancy or fault tolerance in the

processor level; and hence there is a tradeoff between performance and fault tolerance.

Autonomy of processors: Each processor must be autonomous for maximum fault tolerance. There cannot be a *critical* processor whose malfunction could result in system crash.

Communication mechanism: Communication between two processors can be done by shared memory or message transmission or hybrid.

Topology: There cannot be a connection between each processor for practical reasons: topology decides diameter (maximum interprocessor distance) and number of communication paths between two processor and affect performance and the fault tolerance of communication. Routing control is also closely related with performance and fault tolerance.

Reconfiguration: Reshaping of processors must be possible if there are faulty processors.

Extensibility: Eventually when the system expands and needs more processing power: we might ask is it easy to add these processors while still preserving the required fault tolerance, and do we have to design another system from scratch?

3. Definition of Security

The definition of security may be different for different applications. We will define security with military application in mind.

Definition of Security: Computer security is defined as ability of a system (1) to protect information from the unauthorized or accidental access (read, write, or modify), (2) to control inference of information, (3) to control flow of information, and (4) to protect against unauthorized access to communication.

To make more precise definition of protection mechanisms possible, we first set up some risks to computer systems. The risks are [LAND 81] : (1) inference problem, (2) authentication problem, (3) browsing problem, (4) integrity problem, (5) copying, (6) denial of service, (7) confinement problem, (8) Trojan horse, (9) trap door, (10) wiretapping and monitoring, etc.

A system designer has to have a formal model to represent security mechanism, to verify that it is really secure, and to develop a formal methodology to design secure systems.

There are three approaches in modeling security [LAND 81]. In the first approach, security is considered as access control mechanism. Access matrix [LAMP 71] and take-grant [JONE 76] belong to this approach. The second approach models flow of information among objects. Information flow model by Fent [FENT 74] belongs to this approach. The third and last approach view programs as channels for information transfer. Filters [JONE 75] and strong dependency [COHE 77] belong to this model.

Trade-offs between fault-tolerance, security, and performance are discussed in the next section. The following are some of the issues in studying the trade-off.

Architectures: Popular non-Von Neumann architectures are SIMD (single-instruction, multiple-data-stream) computers, MIMD (multiple-instruction, multiple-data-stream) computers, and systolic arrays and wavefront arrays. Obviously the architecture will affect fault tolerance and will affect the design of secure computer systems.

Generalpurpose Security kernel imposes performance degradation, and general purpose kernelized operating systems are 3 to 10 times slower than their counterparts. What is the trade-off between hardware implementation of security and software implementation (for instance security kernel)? And given a fixed number of processors available, then what's the optimum number of security processors?

Extensibility: When we put more processors in the system, is it easy to adapt to new environment while still preserving the previous security or do we have to design the system from scratch and verify the whole system again?

Granularity: How coarse should be the security? Granularity will affect the security mechanism and performance.

Division of Can we use architecture itself as a security mechanism? For instance, each processor is in charge of part of information and each piece of information

may not make sense or may be trivial. The collection of information is very important

Integrated Model Can we develop an integrated model which can represent fault tolerance, security, and performance of a Non-Von Neumann machine?

Metrics of Security: What are good metrics for representing the security? How much security is enough? What's the trade-off between cost and security? What is advantage and disadvantage of Non-Von Neumann architecture in terms of security? If there is any advantage, then how much is it?

Verification: Verification of security is difficult even on a Von Neumann machine. If the design is not well-structured, then it is much more difficult to verify security, especially on Non-Von Neumann machine where many processors are working possibly asynchronously on many processes. Simplified designs make verification easy, but it may cause performance degradation by not fully utilizing the total processing power.

The effect Without fault tolerance, security can not be guaranteed. We can say fault tolerance is precondition of computer security.

4. Tradeoffs between Security and Performance

4.1. Introduction

Efforts to build a high performance secure systems have not been successful so far even though security is very important for many military applications. In this section, we will point out some of the research issues that arise in tradeoffs between security and performance. We first claim that a completely secure high performance system is not feasible with current technology. A decade of research effort in computer security provides us a substantial evidence to support the claim. Some of the problems that have surfaced due to the research are confinement problem[LAMP 73] and trackers problem[DENN 79] which have proven to be extremely difficult. Proving assertions about security has been shown to be undecidable in certain cases[HARR 76]. With such results, we can only hope to get reasonable security with satisfactory performance.

To get a verifiable secure system, it is necessary to use formal modeling techniques. Most models proposed so far have been based on finite state machines. The best known is the Bell and LaPadula model[BELL 73]. Some of the other popular models are High-Water-Mark Model, Lattice Model[BIRK 70], Access Matrix Model[BISH 79] and Information flow Model[FENT 74, DENN 75]. All of these models, however, lack modeling and analysis of performance of the systems. We believe that the first important step in studying the tradeoff between performance and security is the development of formal models of security that incorporate performance considerations. These models should have the analysis capabilities for the following tradeoffs:

(1) Encryption overhead vs Security

Encryption is used frequently to provide security against wire tapping in computer networks or illegal file access in operating systems. Theoretically, the security of the information increases with the size of the key (one time pad being 100% secure). In practice, it is not feasible to have a very long key. Also, the requirements of a particular message may not even require such tight security. Therefore, the formal model should have concepts which take the tradeoffs into consideration.

(2) Validation Overhead vs Security

Researchers have in the past experimented with the idea of a security kernel in an operating system. The main idea behind security kernel is that a small subsystem could validate all accesses to secure objects in the system. However, the results seem to indicate that a high performance security kernel is still not feasible. This again brings out the importance of judging the need for validation.

(3) Granularity of Security vs Performance

The fine granularity of security provides a more flexible control of the information dissemination. For example, in a multilevel secure object the security classification of different parts may be different. Again, the performance considerations make it desirable to have a coarse granularity of security objects, which goes against flexibility of the control.

Our approach to incorporate performance in the security models is to associate timing information with states and state transitions; and using timed Petri

nets. In modeling and simulation for computer systems, existing techniques of queuing theory and graph theory are thus far inadequate, both in terms of their representation ability and their computational tractability. However, it is general consensus that Petri Nets and similar formalisms offer some promise in the modeling of asynchronous, concurrent execution of cooperating processes. The next section describes timed Petri nets. The section following that discusses how they can be used as formal security models and how timing information can be used for the security-performance tradeoff.

As distributed computer systems have become more widely used, interest has increased in techniques and tools which can be used to evaluate their correctness and their performance. In this proposal, we will study techniques for the prediction and verification of security and performance of concurrent systems using Timed Petri Nets.

4.2. Timed Petri Net

Petri nets are abstract and formal graph models for representing the flow of information and control in systems, especially those which exhibit nondeterministic, asynchronous and concurrent properties. Their properties are quite natural and easy to understand, yet their capability of modeling and analyzing such systems are very powerful.

A Petri net is a directed bipartite graph consisting of two types of nodes: *places* and *transitions*. In using graphical representation, they are usually indicated as circles and bars, respectively. Places represent conditions, and transitions represent events. Each transition has certain numbers of input and output places indicating the preconditions and the postconditions of the corresponding event. The holding of the condition in a place is indicated by *token*, which is commonly represented by holding a pattern of tokens in all the places, i.e., a collection of all place markings, which is called a *net marking*. When M is used to denote a net marking, $M(p)$ represents the corresponding marking of place p .

Therefore, a Petri net PN can be formally represented by a 5-tuple (P, T, M_0, I, O) , where

P is the set of places

T is the set of transitions

M_0 is the initial net marking

$I, O: T \rightarrow P^w$ (the power set of P)

$I(t)$ is the set of the input places of transition t

$O(t)$ is the set of the output places of transition t

Petri nets can model dynamic properties of systems. The dynamic behavior is represented by the execution of transitions: this results in changes in systems status. New markings can be obtained from a current marking by following the firing rules of Petri nets. The firing rules are:

- (1) A transition is *enabled from a marking if and only if each of its input places holds at least one token*
- (2) A transition can *fire only if it is enabled*
- (3) After a transition fires, the new marking is obtained by removing one token from each of its input places and adding a token to each of its output places.

If more than one transition is enabled, any one of them may fire, and the choice of which transition to execute is nondeterministic. In the ordinary Petri Net models, the firing of transitions is assumed to be instantaneous. However, in order to evaluate system performance and to model timing characteristics, the notion of *time* is usually included to extend the modeling power of Petri nets. Merlin and Farber extended Petri Nets to include Min times and Max times associated with transitions. Min times define delays during which transitions must remain enabled before they can fire. Max times define maximum delays before a transition must fire. While a transition is enabled, tokens remain on its input places. This permits other transitions, with shorter delays, to rob the transition of its enabling token. This mechanism is particularly useful in modeling timeouts in communication protocols. Merlin and Farber used their Timed Petri Nets to design recoverable systems: that is, systems which can recover from transient failures.

Ramchandani [RAMC' 74] introduced time by associating simple delays with transitions in a Petri Net. Ramamoorthy and Ho then used this Extended Timed Petri Net model to analyze system performance. For a restricted class of systems

that can be modeled using decision free nets, Ramamoorthy [RAMA 80] showed that the performance can be analyzed efficiently. Decision-free nets are a very restricted class of nets which involve neither decisions nor nondeterminism. In decision-free nets, each place can be connected to input of no more than one transition and to the output of no more than one transition. By placing this restriction on the nets, the issue of whether the tokens remain on the input places during the firing delay of a transition becomes irrelevant. Unfortunately, the decision-free restriction is particularly bothersome in modeling communication protocols where decision places are common. Ramamoorthy's work also showed that performance analysis of general Petri Nets is NP-complete. This is indeed discouraging, but it has not, and should not, discourage further work in the area, since many cases have been shown to be easily analyzable.

Zuberek [ZUBE 80] also used timing delays associated with transitions. The restrictions on Petri Nets were relaxed to permit decisions to be modeled. The nets were, however, limited to free-choice nets. In a free-choice net, only one place can be involved in any decision. Zuberek's extensions also required that each transition enabled by a free-choice place be assigned a firing probability. This extension permits the construction of elegant models of lossy transmission media. Zuberek's free choice limitation remains overly restrictive in modeling communication protocols.

Zuberek's definition of time assumes that when a transition is enabled it immediately starts firing by absorbing its input tokens. A transition then continues to fire during its defined delay, and then finishes firing. This is a subtle point which differs from the Merlin and Farber definition of time. Zuberek also introduced an analysis technique based on a *Timed Reachability Graph* (referred to as *TRG*). The *Timed Reachability Graph* (TRG) differs from a reachability in one key aspect: time is introduced as part of the definition of the state of a net. If absolute times are used to describe each state, the reachability graph becomes infinite. Zuberek reduced the state space by using the remaining firing time of currently firing transitions as part of the state component. Therefore, a state in the TRG consists of a marking and a vector of remaining firing times, one for each transition.

A state containing a non-zero remaining firing time (RFT) indicates that the transitions in question are firing while the modeled system is in that state. The TRG can be constructed by systematically calculating successors of each state, starting from the initial state. For a given state, a successor is reached as the result of a transition beginning to fire, or if there are no enabled transitions as the result of a transition finishing firing thereby changing the marking and possibly enabling some transitions. In the later case, time must elapse, and this is accomplished by reducing the remaining firing time of currently firing transitions until one or more transitions finished firing.

The work discussed above deals with deterministic times associated with transitions. Molloy [MOLL 82, MOLL 85] has shown that by assuming exponential distributions of delays, Markov Chain analysis can be used to obtain performance measures. The analysis requires that the untimed reachability graph be constructed. While this work is particularly well suited for modeling systems at a high level of abstraction, it cannot handle fixed timing constraints. Molloy's analysis is based on constructing the un-timed reachability graph and is therefore difficult to use when the graph is large or infinite. In the case of infinite graphs (such as those resulting from models of time-outs), the Petri Net model must be artificially altered to guarantee that the reachability graph is finite.

4.3. Our approach

The Timed Petri net which we propose includes a set of enabling durations(E), a set of firing durations(D), a set of firing frequencies(F), and a set of named resources(R). Each set is associated with the transitions in the net. The model is thus formally defined as follows: The TPN model is a Petri net model which has been augmented to include a set of firing durations(D), a set of firing frequencies (F), and a set of named resources(R). Each set is associated with the transitions in the net. The model is formally defined as follows:

$$TPN=(P,T,A,M',E,D,F,R) \text{ where}$$

$$P=p_1,p_2,\dots,p_n(\text{places})$$

$$T=t_1,t_2,\dots,t_m(\text{transitions})$$

$$A \subseteq P \times T \cup T \times P (\text{directed arcs})$$

$$M' = m_1, m_2, \dots, m_n (\text{initial marking})$$

$$E = e_1, e_2, \dots, e_m (\text{firing durations})$$

$$D = d_1, d_2, \dots, d_m (\text{firing durations})$$

$$F = f_1, f_2, \dots, f_m (\text{firing frequencies})$$

$$R = r_1, r_2, \dots, r_m (\text{resources})$$

where m and n are number of transitions and places, respectively. Each transition in the net must remain enabled for a fixed period, t_e , (its enabling time) before it can fire. A transition is then said to be *firable*, and immediately begins firing by absorbing tokens from its input places. The transition continues to fire for a period, t_f , (its firing duration which we define to be *random variable*). The transition then finishes firing and places tokens on its output places. Enabling times are exactly Merlin's MIN times. Firing times are exactly Zuberek's transition delay. This gives us more powerful modeling power.

Based on our model we will study the following aspects.

- How to get reachability graph for this Petri net?
- What is resource utilization?
- What is mean throughput?
- What is average cycle time for each transition?
- How to cope with state explosion problem?
- Application of our methodology to several concurrent systems.

For reachability analysis we can produce Timed Reachability Graph. The timed reachability graph differs from a conventional reachability graph in one key aspect: *time* is introduced as part of the definition of the state of a Petri net. If absolute times are used to describe each state, then the reachability graph becomes infinite. We can reduce the state space by using the *remaining enabling time and remaining firing time*. Therefore, a state in our timed reachability graph consists of

1. a marking
2. a vector of remaining enabling times
3. a vector of remaining firing times

Resource utilization is the fraction of total time during which a resource is used. It is an one of the important criterion when we compare the performance of concurrent systems.

4.4. Discussion

The Petri Net, which is a transition oriented model, is able to effectively model the flow of control in asynchronous systems. However, they suffer from the state explosion problem. The state explosion problem results from the rapid growth of the size of state transition graphs with increasing system complexity (e.g. communication protocol). For instance, 235 states have been used in [RAZO 80] to model the x.21 interface in an error free environment. But to include undefined and lost signals in the model, 975 states were used. Furthermore, the number of states jumped to 3890 when externally supplied CLEAR signal were included. From this example, we realize that if the number of entities increases and if the services provided by protocol are complex, then the number of states and number of events required for the representation of protocols will increase tremendously and become unmanageable. Therefore, for complex protocols, it is impossible to generate and check all reachable states in a reasonable time or storage. This state explosion problem must be resolved promptly; otherwise the Petri Net model will be intractable for analyzing complex communication protocols in the real world.

In spite of the strength of the Petri Net in representing the control flow, the state explosion problem makes it inadequate for modeling the data transfer aspect of protocols. For instance, it would be difficult to model timers or sequence numbers of messages in communication protocols by employing pure transition oriented models, as a number of states are needed to represent each individual sequence number; for protocols with a large number of sequence number spaces, the problem would be even more serious.

5. Tradeoffs between Fault-Tolerance and Performance

In the past, fault-tolerance [ANDE 81] and performance [FERR 78] in computing systems have been considered separately by most people. But it can be easily understood that there is an interdependency between the two disciplines. Techniques that increase fault-tolerance of the system put an additional workload on the system, which generally results in decreased performance during fault-free operation. Moreover when failures are detected, they initiate countermeasures which increase the load on the system through diagnosis, organizing reconfiguration, and rerun time.

Studies have shown that Cray-1 crashes twice as often as the CMUA, which is an ECL FDP-10 used in the computer science department at Carnegie-Mellon University. But it is also known that Cray-1 can operate continuously at rates above 138 Million Instructions Per Second (MIPS), while the CMUA operates at 1.2 MIPS. Hence Cray-1 executes more than 50 times as many as instructions CMUA between crashes. Inconsistencies like this one is another reason why fault-tolerance modeling and measuring should be closely related with the characterization of the performance of the system under study.

All the existing techniques for achieving the required degree of fault-tolerance are based upon some form of replication. Conceptually, a function from a computer system can be thought of as an execution of some required operations on some data. Therefore to achieve fault-tolerance of a computer system, we need a replication of both execution and data, and this brings out the following trade-off issues to us:

- (1) *Replication of execution vs Performance*: Multiplicity of processors in a parallel machine or a distributed system makes possible the replication of execution as a form of the replication of a process. Many identical processes can be created and can execute in a coordinated way to finish a single task reliably. But as more processes are created for a single task, more computing power is consumed, and this decreases the system performance accordingly. And algorithms to synchronize the processes detect the failing processes, and reconfigure the surviving processes impose additional adversary effect on performance.

- (2) *Replication of data vs Performance*: Storing the identical data on many storage devices increases the availability of the data. As more copies are made, higher availability is achieved. But the management of these multiple copies of data requires algorithms to maintain consistency among the copies in the environment of concurrent updates by many users and network partitions. And these algorithms have more negative impact on the system performance as more copies are made.

While we design and implement an efficient fault-tolerant mechanism with the above trade-off issues in mind, we need a method to describe a proposed mechanism concisely and to check whether this mechanism provides an appropriate level of fault-tolerance and performance. It is well known that formal modeling is a successful technique for these purposes. Although many research efforts have been made in the modeling of fault-tolerance and performance of a computer system, only few of them were intended to model both fault-tolerance and performance.

A first approach to integrated fault-tolerance and performance models is described in [CAST 80]. Realizing that the utilization of a resource exhibits a periodic characteristic and the failure rate of a resource depends on the utilization of that resource, the utilization is modeled by an equation $u(t) = m(t) + z(t)$ where $m(t)$ is a periodic, deterministic function of time and $z(t)$ is a stationary, zero mean, Gaussian process, independent of $m(t)$, and the failure rate is modeled by an equation $\lambda(t) = a u(t) + b$. Using these equations the workload dependent reliability function was obtained. In their approach not only hardware permanent errors but also hardware transient errors and software errors were considered. The model presented in [HAC 83] is based on the fact that user and system demands and their performance have significant influence on calculated reliability measures. Using this model, mean time to failure was calculated as a function of measurable parameters describing workload and system configuration. The classes of failures cover hardware transients and software and hardware design errors and program faults. The model was validated using data monitored on an IBM 4331 installation. In the above two models only workload dependent reliability values were calculated, and reliability-dependent performance values were not considered. In [SCHO 86] a load-dependent fault-tolerance model based on the

Baskett, Chandy, Muntz, and Palacios (BCMP) performance model [BASK 75] was introduced and the issue of how the workload of a system which was characterized by the utilization of each system component affects the failure rate of the system components was addressed. And using the same model the failure effect on workload was explained.

There are some studies [BEAU 78, GAY 79, HUSL 81, MUNA 83, MITR 83, FURC 84] concerning fault-tolerance/performance of Gracefully Degrading Systems. Usually this is done by estimating the performance of each configuration of the system and by considering these configurations as states of a Markovian process with transitions due to failures and repairs. In [GARC 85] an evaluation technique which is useful for studying both the performance and the fault-tolerance of a distributed computing system is presented. The model is based on Markovian process and enables the computation of the average time to successful completion of the user request, taking into account the system failures or repairs which may occur before the request is completed. But some of the problems of the Markov process based techniques are that the system may have many states, making the Markov model difficult to construct, and that the computational complexity of the evaluation technique is very high.

All the works mentioned above are in the context of conventional time-sharing machine and deal with systems with a specific fault-tolerant technique, which is not general enough. To investigate the trade-offs between fault-tolerance and performance of distributed system and parallel machines, more powerful modeling techniques should be sought which can handle important features in the concurrent systems, like various execution strategies (e.g., load balancing, selecting a certain copy from several replications, etc) and many different interconnection or communication networks (e.g., hypercube, omega network, shuffle-exchange network, bus, partially connected network, etc). Also the new model should be able to explain a wide range of fault-tolerant techniques. Efforts have been made to develop a unified approach to reliability modeling of fault-tolerant computers which is both general and practical [NG 80]. We believe that study in this unified reliability modeling will be helpful in the quest of more general fault-tolerance/performance modeling.

Although modeling is a technique with which we can represent a complex system in a very concise way, in many cases it tends to oversimplify the system under investigation, and many minor details are lost because its expressive power is limited; and we also have to consider the computational tractability of the resulting model. Therefore we need to check the validity of the simplified model and also need to improve its expressive power without increasing too much computational complexity. A technique which satisfies these two requirements is simulation. A model can be validated by comparing its result with that of the simulation. And the combination of modeling and simulation can represent the system more correctly, with reasonable computational requirements, by capturing the backbone of the system in the model, representing minor details in the simulation, and coupling these two in a hierarchical fashion. Therefore we believe that both modeling and simulation technique should be utilized in the research of tradeoffs between fault-tolerance and performance.

6. Tradeoffs between Security and Fault Tolerance

Many computer applications, e.g. databases for commercial or strategic information, control program for factory control or battle management, need high reliability, fault-tolerance and security. At the same time the mechanisms to provide either security or fault-tolerance tend to be quite expensive in terms of overhead. The security and fault-tolerance problems have many characteristics in common [RAND 86]. Hence it is very desirable (i) to identify common subproblems between fault tolerance and security, (ii) to have common mechanism to support both security and fault-tolerance, reducing total overhead, and (iii) to study the trade-offs between the degree of fault tolerance and the degree of security, for given constraints of overhead.

As suggested by [LAPR 85], we can view security and fault-tolerance as two different aspects of a common problem. Some of the security problems can be viewed as design errors or a failure of the system. For example, denial of service on a radio channel by jamming is a breach of security; and at the same time it is a system failure, namely lack of communication. We can think of this situation as an unanticipated error in the design, since by choosing random frequencies for the channel, this problem can be almost eliminated.

We may treat many of security problems and faults as unanticipated failures, which can be reduced by following methodologies like dual design [RAMA 81]. During the design phase also, there are some mechanisms to be used for achieving both fault tolerance and security. For example a secure system might be designed using internal security checks, for example based on access matrix, which will contain some security violations, just as for fault tolerance, the recovery block scheme uses acceptance tests. Another example may be the following- in a distributed system we may encrypt a message after partitioning it into n parts, such that the receiver needs exactly m ($m < n$) parts to decode the message. We try to send the parts of one message via different paths to the destination. This mechanism protects us against the loss of packets and transmission error, at the same time it provides security against wire tapping.

There are some differences in security and fault-tolerance. The mechanism to achieve one might make it more difficult to ensure the other. For example, one maintains multiple copies of datafiles on network to improve fault-tolerance, yet this leads to poor security, as we have to guard more locations to ensure the security of the datafile. While designing the system, we can have some trade-off between the degree of security and fault tolerance. For example the computational overhead constraint might force to give more computation cycles to fault tolerance compared to security. We realize, finally, that the similarities and the differences between security and fault tolerance makes this problem unique, significant, as well as topical in the specialized computer system and network today.

7. REFERENCES

- [ANDE 81] Anderson, T. and Lee, P.A., *Fault Tolerance Principles and Practices*, Prentice-Hall International, London, 1981.
- [BASK 75] Baskett, F., Chandy, K.M., Muntz, R.R., and Palacios, F.G., "Open, closed, and Mixed Networks of Queues with Different Classes of Customers," *Journal of ACM*, vol. 22, no. 2, pp. 248-260, April 1975.
- [BEAU 78] Beaudry, M.D., "Performance-Related Reliability Measures for Computing Systems," *IEEE Trans. on Computers*, vol. C-27, no. 6, pp. 540-547, June 1978.
- [BELL 73] Bell, D.E. and LaPadula, L.J., "Secure Computer Systems: Mathematical Foundations," ESD-TR-73-278, vol. 1, ESD/AFSC, Hanscom AFB, Bedford, Mass., Nov. 1973/
- [BIRK 70] Birkhoff, G. and Bartee, T.C., *Modern Applied Algebra*, p.260, McGraw-Hill, New York, 1970.
- [BISH 70] Bishop, M. and Snyder, L., "The Transfer of Information and Authority in a Protection System," *Proc. 7th Symp. Operating Systems Principles*, pp.45-54, Dec. 1979.
- [CAST 80] Castillo, X. and Siewiorek, D.P., "A Performance-Reliability Model for Computing Systems," *IEEE 1980 Fault-Tolerant Computing*, pp. 187-192, 1980.
- [COHE 77] Cohen, E., "Information transmission in computational systems," *Proceedings of 6th symp. Operating Systems Principles*, ACM SIGOPS Operating Syst. Rev., Vol. 11, No. 5 (Nov. 1977), 133-140.
- [DENN 75] Denning, D.E., "Secure Information Flow in Computer Systems," Ph.D. dissertations, Purdue University, May 1975.
- [DENN 79] Denning, D.E., Denning, P.J., and Schwartz, M.D., "The Tracker: A Threat to Statistical Database Security", *ACM Trans. Database Syst.*, vol. 4, no. 1, pp.76-96, March 1979.

- [FENT 74] Fenton, J.S., "Memoryless subsystems," *Comput. J.*, Vol. 17, No. 2 (May 1974), 143-147.
- [FERR 78] Ferrari, D., *Computer Systems Performance Evaluation*, Prentice Hall, 1978.
- [FURC 84] Furchtgott, D.G. and Meyer, J.F., "A Performability Solution Method for Degradable Nonrepairable Systems," *IEEE Trans. on Computers*, vol. 33, no. 6, pp. 550-554, June 1984.
- [GARC 85] Garcia-Molina, H. and Kent, J., "Evaluating Response Time in a Faulty Distributed Computing System," *IEEE Trans. on Computers*, vol. 34, no. 2, pp. 101-109, February 1985.
- [GAY 79] Gay, F.A. and Ketelsen, M.L., "Performance Evaluation for Gracefully Degrading Systems," *IEEE 1979 Fault-Tolerant Computing*, pp. 51-58, 1979.
- [HAC 83] Hac, A., "A Performance Considered Model for System Reliability," *IEEE 1983 Fault-Tolerant Computing*, pp. 139-143, 1983.
- [HARR 76] Harrison, M.A., Ruzzo, W.L., and Ullman, J.D., "Protection in Operating Systems," *CACM*, vol. 19, no. 8, pp.461-471, Aug. 1976.
- [HUSL 81] Huslende, R., "A Combined Evaluation of Performance and Reliability for Degradable Systems," *ACM-SIGMETRICS Conf. on Measurement on Modeling of Comp.*, pp. 157-164, 1981.
- [JONE 75] Jones, A.K. and Lipton, R.J., "The enforcement of security policies for computation," *Proc. 5th Symp. Operating Systems Principles*, *ACM SIGOPS Operating Systems Rev.*, Vol. 9, No. 5 (Nov. 1975), 197-206.
- [JONE 76] Jones, A.K., Lipton, R.J. and Snyder, L., "A linear time algorithm for deciding subject-object security," *Proc. 17th Ann. Foundations Computer Sci. Conf.*, Houston, Tex., 1976, pp. 33-41.
- [LAMP 71] Lampson, B.W., "Protection," in *Proc. 5th Princeton Symp. Information Sciences and Systems*, Mar. 1971, pp. 437-443.
- [LAMP 73] Lampson, B.W., "A Note on the Confinement Problem," *CACM*, vo. 16, no. 10, pp.613-615, Oct. 1973.

- [LAND 71] Landwehr, C.E., "Formal Models for Computer Security," ACM Computing Surveys, Vol. 13, No. 3 Sep. 1981, pp. 247-278.
- [LAPR 85] Laprie, J.C., "Dependable Computing and Fault Tolerance," Proc. FTCS-15, June 1985.
- [MITR 83] Mitrani, I. and King, P.J.B., "Multiserver Systems Subject to Break-downs: An Empirical Study," IEEE Trans. on Computers, vol. 32, no. 1, pp. 96-98, January 1983.
- [MOLL 82] Molloy, M.K., "Performance modeling using stochastic Petri nets," IEEE Trans. Comput., vol. C-31, pp. 913-917, Sept. 1982.
- [MOLL 85] Molloy, M.K., "Discrete time stochastic Petri nets," IEEE Trans. Software Eng., vol. SE-11, pp. 417-423, Apr. 1985.
- [MUNA 83] Munarin, J.A., "Dynamic Workload Model for Performance/Reliability Analysis of Gracefully Degrading Systems," IEEE 1983 Fault-Tolerant Computing, pp. 290-295, 1983.
- [NG 80] Ng, Y.W. and Avizienis, A.A., "A Unified Reliability Model for Fault-Tolerant Computers," IEEE Trans. on Computers, vol. C-29, no. 11, pp. 1002-1011, November 1980.
- [RAMA 80] Ramamoorthy, C.V. and Ho, G.S., "Performance evaluation of asynchronous concurrent systems using Petri nets," IEEE Trans. Software Eng., vol. SE-6, pp 440-449, Sept. 1980.
- [RAMA 81] Ramamoorthy, C.V., Mok, Y.R., Bastani, F.B., Chin, G.E., and Suzuki, K., "Application of a Methodology for the Development and Validation of reliable Process Control Software," IEEE Trans. on Software Engineering, Vol. SE-7, No. 6, November 1981.
- [RAMC 74] Ramchandani, C., "Analysis of asynchronous concurrent systems by timed Petri nets," Ph.D. dissertation, Massachusetts Inst. of Technol., Cambridge, Project Mac Rep. MAC-TR-120, 1974
- [RAZO 80] Razouk, R. and Estrin, G., "Modeling and Verification of Communication Protocols in SARA : The X.21 Interface", IEEE Trans. Computer., Vol C-29, No 12, Dec 1980.

- [RAND 86] Randell, B. and Dobson, J. E., "Reliability and Security Issues in Distributed Computing Systems," University of Newcastle upon Tyne.
- [SCHO 86] Schoen, O., "On a Class of Integrated Performance/Reliability Models based on Queueing Networks," IEEE 1986 Fault-Tolerant Computing, pp. 90-95, 1986.
- [SIEW 82] Siewiorek, D.P. and Swarz, R.S., *The Theory and Practice of Reliable System Design*, Digital Press, Bedford, Mass., 1982.
- [ZUBE 80] Zuberek, W.M., "Timed Petri nets and preliminary performance evaluation," in *Proc. 7th Annu. Symp. Compt. Architecture*, 1980, pp. 88-96.



MISSION of *Rome Air Development Center*

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control, Communications and Intelligence (C³I) activities. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C³I systems. The areas of technical competence include communications, command and control, battle management information processing, surveillance sensors, intelligence data collection and handling, solid state sciences, electromagnetics, and propagation, and electronic reliability/maintainability and compatibility.